System Verification and Validation Plan for RoCam

Team #3, SpaceY
Zifan Si
Jianqing Liu
Mike Chen
Xiaotian Lou

October 26, 2025

Revision History

Date	Version	Notes
Date 1	1.0	Initial Draft

Contents

1 Symbols, Abbreviations, and Acronyms

symbol	description
Yolo	You only look once – an existing object detection model
CNN	Convolutional Neural Network
Jetson	abbreviatio for specifcally Jetson Nano Orin, which is the proposed board for this project
STM32	abbreviation for specifically STM32 microcontroller, which is the proposed microcontroller

This Verification and Validation (V&V) plan outlines how the RoCam project will demonstrate that the system is built correctly (verification) and fits stakeholder needs (validation). It anchors testing to the SRS, architecture (MG), and detailed design (MIS), ensuring every requirement is traceable to one or more tests. Verification relies on structured reviews, checklists, and staged testing (unit, integration, and system) to confirm functional behavior, performance expectations, interface consistency, and safety assumptions. Validation complements this with stakeholder walkthroughs, task-based inspections, and controlled field or demo sessions to confirm the system's real-world suitability and operator workflow. Roles and responsibilities are assigned to team members and the supervisor to keep eviews focused and accountable, with feedback captured and tracked. Priorities emphasize correctness, performance, reliability, and safety; lower-priority or out-of-scope items (e.g., exhaustive usability studies or independent verification of third-party components) are noted to keep the plan feasible. The document concludes with traceability tables, placeholders for unit-level details after design finalization, and appendices for parameters and optional surveys—forming a clear, practical roadmap for building credible evidence of quality.

TODO: tidy up deliverables list

2 General Information

2.1 Summary

The RoCam system is a ground-based camera and gimbal assembly designed to autonomously track model rockets during launch and flight. It utilizes a high-resolution camera coupled with a motorized gimbal to maintain visual lock on the rocket, providing real-time data and video feed. The system consists of a STM32 based motion controller that interfaces with the physical gimbal; a Jetson compute module that processes the camera feed; and a react frontend that displays the video preview and allows the user to control the gimbal. The primary purpose of RoCam is to replace the traditional unreliable manual tracking methods with an automated solution to produce high-quality flight data and video footage. To better assist model rocket teams across Canada to conduct successful launches by providing an affordable and reliable system that can be easily track launching movements.

2.2 Objectives

The primary objective of this Verification and Validation V&V Plan is to establish confidence in the correctness, performance, and reliability of the RoCam system by ensuring that all verification and validation activities are thorough, traceable, and aligned with project requirements. The plan emphasizes testing and peer review to confirm that functional and performance criteria are fully addressed, with improvements tracked through GitHub Issues. Validation focuses on both automated and field testing to confirm that the system performs as intended in real-world launch conditions. These qualities—correctness, performance, reliability, and traceability—are fundamental to the success of the V&V process, ensuring that the RoCam system is both technically sound and operationally dependable. Certain objectives, such as independent verification of third-party hardware and extended

usability testing, are excluded due to scope and resource limitations, with reliance placed on manufacturer and stakeholder validation.

2.3 In-Scope Objectives

Verification and validation efforts will prioritize the following qualities:

- **Correctness:** Ensure the reliability of tracking and control algorithms through testing and validation.
- **Performance:** Achieve real-time processing of 1080p 60 fps video and maintain minimal gimbal control latency.
- **Safety:** Guarantee that the system operates safely under assumed conditions, with fail-safe mechanisms.

2.4 Out-of-Scope Objectives

The following objectives are excluded from this project due to time, budget, and resource constraints:

- **Comprehensive Usability Testing:** User experience will be evaluated informally via team and client feedback rather than large-scale formal studies.
- **Third-Party Component Verification:** External hardware such as the gimbal controller and camera are assumed to perform according to their manufacturer specifications.
- **Scalability Beyond Small-Scale Rockets:** Tracking a bullet or extremely difficult targets beyond a model rocket is outside the scope of this project.

By defining these priorities and exclusions, the RoCam project ensures that available effort is directed toward validating the most critical system qualities: **performance, accuracy, and operational safety**.

2.5 Extras

Extras

Beyond the core objectives, the RoCam project includes several extras that enhance its quality and usability:

- **STM32 control circuit design:** Development of a custom control circuit to interface the gimbal actuators with the host computer, enabling precise motion control and feedback sensing.
- **User Instructional Video:** Creation of a short instructional video demonstrating system setup, calibration, and operation to support new users and ensure safe deployment.

These extras aim to improve the overall usability, accessibility, and documentation quality of the final deliverable, reinforcing RoCam's emphasis on practical integration between computer vision software, mechanical control, and end-user experience.

2.6 Relevant Documentation

Software Requirements Specification (SRS Author (2019)): The SRS document serves as a outline for the VnV plan. The SRS document contains specific functional and non-functional requirements, performance criteria, and constraints. The SRS document also contains detailed stakeholder roles and scope of the work. Aligning the VnV plan with the SRS document ensures that all specified requirements are adequately verified and validated.

Software Architecture MG (MG ?) The MG document provides details of our system architecture. To effectively design tests to validate and verify the system, it is important to understand the architecture and functionality of each module.

Software Detailed Design Document (MIS ?): The MIS document provides detailed design of our system. It provides critical information about the implementation details, including data structures, algorithms, and interfaces. This document is essential for efficient test design to specific modules.

3 Plan

This section outlines the overall Verification and Validation (V&V) plan for the **RoCam** project. It defines the structure, responsibilities, and methods used to verify that the system meets its functional and non-functional requirements and to validate that it fulfills stakeholder needs. The section covers the verification of the SRS, design, implementation, and automated testing processes, as well as the validation strategy through stakeholder reviews and field testing. Its purpose is to provide a clear, organized framework ensuring that all aspects of RoCam are systematically tested, reviewed, and proven reliable before final deployment.

3.1 Verification and Validation Team

Table 1: Verification and Validation Team Responsibilities

Name	Role	V&V Responsibilities
Zifan Si	Front End Interface Tester	Responsible for verifying and validating the usability and functionality of the react frontend interface. Ensure usability, robustness of frontend features.
Jianqing Liu	Team Lead	Conduct integration test and system validation. Validate the functionality and performance of the system against requirements and use cases. Validate system design and functionality.
Mike Chen	CV Tester	Verify and validate computer vision pipeline performance, including object detection and tracking accuracy.
Xiaotian Lou	Integration Tester and test automation developer	Implement automated testing frameworks in all software components. Implement API, and integration test across different modules.
Dr. Shahin Sirouspour	Faculty Supervisor	Provides oversight and ensures that verification and validation procedures align with academic and engineering standards. Reviews test methodologies, validates safety-critical procedures, and confirms that deliverables meet course and institutional requirements.

3.2 SRS Verification

The SRS verification will focus on verify whether the system design satisfy all functional and non-functional requirements, stakeholder needs, scope of work and constraints specified. The SRS verification will be conducted through the following steps:

3.2.1 functional and Non-Functional Requirement Review

Since the Design function will evaluate in detail whether the system align with the functional and non-functional requirements. The SRS verification will focus on verify the system design as a whole against the requirements. It will be implemented as integration test in the final stages of development.

3.2.2 Stakeholder Review

Several review sessions will be schedules with the stakeholders, such as Dr. Shahin Sirouspour, the McMaster Rocketry Team. Our team can revise the future feature implementation

based on their feedback.

3.2.3 Constraints and Scope of Work Review

Several internal review sessions will be conducted to ensure that our system design and implementation are within the defined scope of work and constraints specified in the SRS document.

Table 2: SRS Verification Checklist

Phase	**Verification Activities**
Functional & Non-Functional Requirement Review	☐ Review all requirements against current system design ☐ Create automated system tests for functional requirements ☐ Measure performance, safety, and reliability against non-functional requirements ☐ Record test results and address any failed cases ☐ Update traceability matrix after verification
Stakeholder Review	☐ Schedule review sessions with stakeholders ☐ Present verified results and collect feedback ☐ Document required changes based on feedback ☐ Obtain stakeholder confirmation or approval
Constraints & Scope of Work Review	☐ Verify implementation stays within project scope ☐ Check compliance with all technical and safety constraints ☐ Review design against timeline and budget limits ☐ Update records if scope or constraints change
Integration Verification	☐ Conduct integration tests covering all system components ☐ Validate full system behavior against SRS requirements ☐ Log and resolve all integration issues ☐ Prepare final verification report for stakeholder review

3.3 Design Verification

The design verification process for the **RoCam** project ensures that the proposed architecture, algorithms, and hardware interfaces satisfy all functional and non-functional re-

quirements before implementation. Design verification activities will be conducted through structured design reviews, peer evaluations, and the use of detailed verification checklists. The goal is to confirm that the design meets the intended performance, reliability, and safety objectives defined in the SRS.

Verification Plan

The design verification will proceed in three main stages:

1. **Internal Design Review:** Each subsystem (computer vision, control, and UI) will undergo an internal review by each team member. Each member must agree on the designed architecture, algorithms, and interfaces. Interfaces must be clearly defined for scalability.

2. **Peer Review by Classmates:** A peer design review session will be conducted with other capstone teams to obtain external feedback. Reviewers will evaluate the system's clarity, feasibility, and maintainability based on the shared design artifacts (block diagrams , interface specifications, and test plans).

3. **Supervisor and TA Review:** The project supervisor and teaching assistants will review the finalized design to ensure compliance with academic standards, capstone expectations, and safety considerations.

Verification Artifacts

Design verification will produce the following artifacts:

- Verified design diagrams and interface specifications.
- Annotated review feedback forms from classmates and supervisor.
- Updated design documents reflecting corrections and improvements.
- Completed design verification checklists (see below).

Design Verification Checklist

The following checklist will be used during design reviews to ensure completeness and consistency across all modules:

Table 3: Design Verification Checklist

Phase	**Verification Activities**
Internal Design Review	☐ Review subsystem architectures (vision, control, UI) with all team members ☐ Confirm APIs that are agreed upon ☐ Verify all module interfaces are clearly defined and documented in code ☐ Ensure design supports scalability and modularity ☐ Record review comments and update design diagrams
Peer Review by Classmates	☐ Present system design and interface specifications to peer teams ☐ Collect peer feedback on clarity, feasibility, and maintainability ☐ Address identified issues or ambiguities in documentation ☐ Record peer feedback summaries and resolutions
Supervisor and TA Review	☐ Submit finalized design package (diagrams, specs, test plans) for review ☐ Confirm compliance with academic and capstone standards ☐ Review safety considerations and risk mitigations ☐ Document supervisor and TA comments and update design documents accordingly
Verification Artifacts	☐ Store verified design diagrams and updated interface specifications ☐ Archive annotated feedback forms from peers and supervisor ☐ Maintain an updated version-controlled design document repository ☐ Complete and file the final design verification checklist

By following this structured verification plan and checklist, the team ensures that the RoCam design is both technically sound and fully aligned with its performance and safety goals prior to implementation.

3.4 Verification and Validation Plan Verification

The verification of the verification and validation (VnV) plan for the **RoCam** project will be verified through testing and peer review from peers and supervisors. Each iteration of

feedbacks and improvements will be tracked with github issues. The focus on the verification plan will ensure functionality, performance metrics are adequately covered in design.

For the validation plan, it will be measured through a combination of field tests and automated test cases. The automated test will make sure our program functions as expected in the developing environment. The field test will be used as our integration tests to validate the entire system.

Table 4: RoCam Verification and Validation (V&V) Checklist

Phase	**Actionable Verification and Validation Activities**
Verification Plan Review	☐ Conduct internal reviews of the V&V plan with team and supervisor ☐ Ensure functionality and performance metrics are fully covered ☐ Perform peer review sessions for completeness and clarity ☐ Track feedback and revisions using GitHub Issues ☐ Confirm all review comments are resolved before approval
Automated Verification Testing	☐ Develop automated tests for vision, control, and UI components ☐ Verify software behavior under simulated conditions ☐ Record test results and compare against performance criteria ☐ Update test scripts based on identified defects or regressions
Field Validation Testing	☐ Conduct full system integration tests during field trials ☐ Validate real-time tracking, communication, and safety functions ☐ Collect and analyze field performance data (latency, accuracy, stability) ☐ Log and resolve issues through GitHub and retest after fixes
Continuous Monitoring	☐ Review automated and field test outcomes after each iteration ☐ Maintain test evidence and validation summaries in repository ☐ Update the V&V plan to reflect system improvements and new test cases

3.5 Implementation Verification

The implementation verification for the **RoCam** project ensures that the developed software and integrated hardware components correctly realize the verified design. This phase confirms that the implemented system satisfies the functional and non-functional requirements defined in the SRS through a combination of structured testing, static verification, and peer review.

Dynamic Verification

Dynamic verification activities focus on confirming that the system behaves as intended under realistic operating conditions. They include the following elements:

- **Unit Testing:** Each software module (vision processing, gimbal control, data recording, and UI management) will undergo unit testing to verify functional correctness in isolation.
- **Integration Testing:** Once individual modules are verified, integration tests will confirm proper data flow between subsystems (camera input $\rightarrow$ vision algorithm $\rightarrow$ gimbal control $\rightarrow$ UI). Tests will ensure synchronization, timing correctness, and robustness under real-time constraints.
- **System Testing:** Full-system field tests will validate end-to-end performance, including real-time video tracking, system state transitions (Idle, Armed, Tracking), and video recording accuracy at 1080p 60 fps. These tests correspond directly to the verification items in the SRS functional and performance requirements.

Static Verification

Static verification will be performed throughout the implementation phase to ensure that the codebase meets quality, safety, nd maintainability standards before execution. Planned techniques include:

- **Code Walkthroughs:** Conducted during the final CAS 741 presentation, where each subsystem lead presents core implementation logic and design decisions. This serves as a semi-formal walkthrough for peer and instructor review, enabling early identification of potential design or coding issues.
- **Code Inspection:** Team members will review each other's pull requests prior to merging into the main branch. Reviews will focus on coding standard compliance, potential logic errors, exception handling, and safety-critical sections (e.g., gimbal actuation and state transitions).
- **Static Analysis Tools:** Tools such as `clang-tidy`, `cppcheck`, and `pylint` will be used to detect memory leaks, uninitialized variables, and other common implementation defects. Results from static analysis will be logged and reviewed as part of the verification record.

- **Documentation Review:** The team will verify that all inline comments, module-level documentation, and API references are consistent with the design specification and user documentation requirements.

Presentation and Usability Review

The final CAS 741 presentation will serve as both a **code walkthrough** and a partial **usability evaluation**. During the presentation, peers and instructors will observe live system operation, assess user interface clarity, and provide feedback on usability, responsiveness, and overall system behavior. This qualitative feedback will be incorporated into the final verification summary and used to refine user-facing components prior to final submission. Together, these verification activities ensure that the RoCam implementation is thoroughly tested, statically verified, and validated against both its design and real-world performance expectations.

3.6 Automated Testing and Verification Tools

The RoCam project adopts a structured automated testing approach that prioritizes maintainability, integration compatibility, and developer productivity. The selection of testing frameworks was guided by three key criteria: (1) **ecosystem compatibility**, ensuring each tool aligns with its respective language and runtime; (2) **ease of automation**, to integrate smoothly with CI/CD pipelines; and (3) **scalability**, so that test coverage can expand as the project evolves. These principles ensure a consistent and reliable testing environment across backend, frontend, and end-to-end validation stages.

Each subsystem of RoCam employs tools optimized for its function and implementation language: **Testing Frameworks:**

- **Backend (Python):** The project uses **Pytest** for unit testing due to its simplicity, readability, and powerful plugin system. It supports fixture management and integrates easily with coverage analysis and mock testing.
- **Frontend (React/TypeScript):** The project uses **Jest**, which is tightly integrated with React and TypeScript. It enables rapid component testing, snapshot validation, and ensures that UI logic behaves as expected across updates.
- **System-Level and Web UI:** The project uses **Playwright** for end-to-end (E2E) and Web UI testing. Playwright enables simulation of realistic operator workflows such as arming, tracking, and recording, ensuring smooth coordination between the frontend, backend, and physical hardware layers.

Code Coverage and Reporting Plan:
Coverage analysis is conducted using the following tools:

- **pytest-cov:** Provides granular coverage reporting for Python code, including line, branch, and function-level metrics, seamlessly integrated with Pytest test runs.

- **Jest (built-in coverage):** Tracks statement, branch, and function coverage for TypeScript components and produces standardized reports that align with Python output formats for unified tracking.

Linters and Formatters:
To maintain coding standards and prevent stylistic or logical errors, RoCam enforces consistent linting and formatting across both stacks:

- **Python:** Uses **Ruff** and **Black** for linting and code formatting to enforce PEP 8 compliance and maintain readability.
- **Frontend (TypeScript):** Uses **ESLint** and **Prettier** to maintain uniform style rules and detect syntax or logical inconsistencies.

3.7 Software Validation

Software validation confirms that **RoCam** satisfies stakeholder needs and accurately implements the intended use cases ("building the right system"). Validation complements verification and is referenced to the SRS goals nd stakeholder roles.

Stakeholder Reviews and Task-Based Inspection

- **Requirements Walkthroughs:** Scheduled reviews of the SRS and UI wireframes with the McMaster Rocketry Team (client) to confirm scope, priorities, and acceptance criteria.
- **Task-Based Inspection:** Stakeholders perform representative tasks (arm system, initiate tracking, stop tracking, export footage) while observers record issues, ambiguities, and unmet expectations.
- **Peer Reviews:** Classmate reviews focus on clarity, testability, and completeness (per CAS 741 guidance); all feedback and resolutions are logged.

Field and Demo-Based Validation

- **Rev 0/PoC Demo:** Used to validate that early functionality aligns with stakeholder expectations (e.g., preview latency, arm/track state semantics). Post-demo debrief captures change requests.
- **Supervisor Validation:** A focused session shortly after Rev 0 to confirm that the demonstrable behaviors match the SRS's product use cases and performance intent.
- **User (Operator) Sessions:** "Hallway" usability checks with camera operators validate readability in outdoor-like conditions and UI flow (single-page ops without scrolling on 1920×1080).

External Data and Bench Validation

- **Reference Footage:** Where live launches are unavailable, public model-rocket videos and internally recorded dry-runs are used to validate detection/tracking logic and reacquisition behavior.
- **Synthetic Scenarios:** Scripted clips with occlusion/smoke/backlight and varying apparent rocket sizes validate robustness claims prior to field tests.

Acceptance and Traceability

- **Acceptance Criteria:** Derived from the SRS (e.g., sustained 1080p 60 fps I/O, end-to-end latency ≤ 120 ms, successful tracking through nominal flight phases, recording with a ngle overlays).
- **Traceability:** Validation tests map to SRS goals and stakeholder needs; a table links each validation activity to the corresponding SRS item(s) and planned evidence (videos, logs, checklists).

If no suitable external datasets are available for certain scenarios, this limitation will be stated explicitly; in those cases, validation will rely on stakeholder task-based inspections and controlled synthetic footage. This section also references the SRS verification section for consistency checks between validated behaviors and specified requirements.

4 System Tests

4.1 Tests for Functional Requirements

4.1.1 Area of Testing1

Title for Test

1. test-id1

   Control: Manual versus Automatic

   Initial State:

   Input:

   Output:

   Test Case Derivation:

   How test will be performed:

2. test-id2

   Control: Manual versus Automatic

   Initial State:

Input:

Output:

Test Case Derivation:

How test will be performed:

4.1.2 Area of Testing2

...

4.2 Tests for Nonfunctional Requirements

4.2.1 Area of Testing1

Title for Test

1. test-id1

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input/Condition:

   Output/Result:

   How test will be performed:

2. test-id2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

4.2.2 Area of Testing2

...

4.3 Traceability Between Test Cases and Requirements

5 Unit Test Description

5.1 Unit Testing Scope

5.2 Tests for Functional Requirements

5.2.1 Module 1

1. test-id1

   Type:
   Initial State:
   Input:
   Output:
   Test Case Derivation:
   How test will be performed:

2. test-id2

   Type:
   Initial State:
   Input:
   Output:
   Test Case Derivation:
   How test will be performed:

3. ...

5.2.2 Module 2

...

5.3 Tests for Nonfunctional Requirements

5.3.1 Module ?

1. test-id1

   Type:
   Initial State:

Input/Condition:

Output/Result:

How test will be performed:

2. test-id2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

5.3.2 Module ?

...

5.4 Traceability Between Test Cases and Modules

References

Author Author. System requirements specification. https://github.com/..., 2019.

6 Appendix

This is where you can place additional information.

6.1 Symbolic Parameters

The definition of the test cases will call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.

6.2 Usability Survey Questions?

Appendix — Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Lifelong Learning.

The purpose of reflection questions is to give you a chance to assess your own learning and that of your group as a whole, and to find ways to improve in the future. Reflection is an important part of the learning process. Reflection is also an essential component of a successful software development process.

Reflections are most interesting and useful when they're honest, even if the stories they tell are imperfect. You will be marked based on your depth of thought and analysis, and not based on the content of the reflections themselves. Thus, for full marks we encourage you to answer openly and honestly and to avoid simply writing "what you think the evaluator wants to hear."

Please answer the following questions. Some questions can be answered on the team level, but where appropriate, each team member should write their own response:

1. What went well while writing this deliverable?

2. What pain points did you experience during this deliverable, and how did you resolve them?

3. What knowledge and skills will the team collectively need to acquire to successfully complete the verification and validation of your project? Examples of possible knowledge and skills include dynamic testing knowledge, static testing knowledge, specific tool usage, Valgrind etc. You should look to identify at least one item for each team member.

4. For each of the knowledge areas and skills identified in the previous question, what are at least two approaches to acquiring the knowledge or mastering the skill? Of the identified approaches, which will each team member pursue, and why did they make this choice?